嘟嘟独立博客

爱生活爱编码

search...

文章目录 隐藏目录

Spring Boot干货系列：（五）开发Web应用之JSP篇

Spring Boot干货系列 Spring Boot

关闭阅读模式

2017-03-13

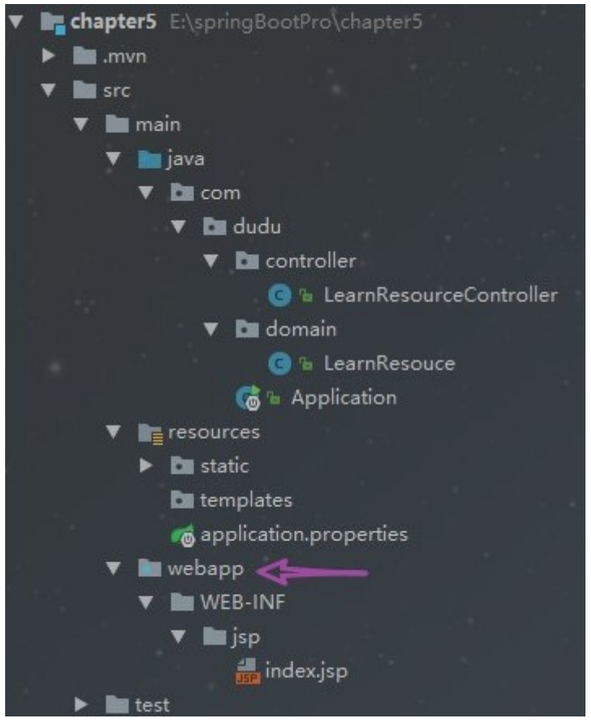
前言

上一篇介绍了Spring Boot中使用Thymeleaf模板引擎，今天来介绍一下如何使用SpringBoot官方不推荐的jsp,虽然难度有点大，但是玩起来还是蛮有意思的。

正文

先来看看整体的框架结构，跟前面介绍Thymeleaf的时候差不多，只是多了webapp这个用来存放jsp的目录，静态资源还是放在resources的static下面。

查看 15 条评论



引入依赖

```
<!--WEB支持-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!--jsp页面使用jstl标签-->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>

<!--用于编译jsp-->
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```

使用内嵌的tomcat容器来运行的话只要这3个就好了。这里介绍下maven中scope依赖范围的概念，因为后续涉及到这个会有问题。

依赖范围就是用来控制依赖和三种classpath(编译classpath，测试classpath、运行classpath)的关系，Maven有如下几种依赖范围：

- **compile:**编译依赖范围。如果没有指定，就会默认使用该依赖范围。使用此依赖范围的Maven依赖，对于编译、测试、运行三种classpath都有

- **test:** 测试依赖范围。使用次依赖范围的Maven依赖，只对于测试classpath有效，在编译主代码或者运行项目的使用时将无法使用此依赖。典型的例子是Jnuit,它只有在编译测试代码及运行测试的时候才需要。
- **provided:**已提供依赖范围。使用此依赖范围的Maven依赖，对于编译和测试classpath有效，但在运行时候无效。典型的例子是servlet-api，编译和测试项目的时候需要该依赖，但在运行项目的时候，由于容器以及提供，就不需要Maven重复地引入一遍。

—— application.properties配置 ——

要支持jsp，需要在application.properties中配置返回文件的路径以及类型

```
spring.mvc.view.prefix: /WEB-INF/jsp/
spring.mvc.view.suffix: .jsp
```

这里指定了返回文件类型为jsp,路径是在/WEB-INF/jsp/下面。

—— 控制类 ——

上面步骤有了，这里就开始写控制类，直接上简单的代码，跟正常的springMVC没啥区别：

```
@Controller
@RequestMapping("/learn")
public class LearnResourceController {
    @RequestMapping("")
    public ModelAndView index(){
        List<LearnResouce> learnList =new ArrayList<LearnResouce>();
        LearnResouce bean =new LearnResouce("官方参考文档","Spring Boot Reference Guide","http://docs
        learnList.add(bean);
        bean =new LearnResouce("官方SpriongBoot例子","官方SpriongBoot例子","https://github.com/spring
        learnList.add(bean);
        bean =new LearnResouce("龙国学院","Spring Boot 教程系列学习","http://www.roncoo.com/article/de
        learnList.add(bean);
        bean =new LearnResouce("嘟嘟MD独立博客","Spring Boot干货系列 ","http://tengj.top/");
        learnList.add(bean);
        bean =new LearnResouce("后端编程嘟","Spring Boot教程和视频 ","http://www.toutiao.com/m15590967
        learnList.add(bean);
        bean =new LearnResouce("程序猿DD","Spring Boot系列","http://www.roncoo.com/article/detail/12
        learnList.add(bean);
        bean =new LearnResouce("纯洁的微笑","Sping Boot系列文章","http://www.ityouknow.com/spring-boot
        learnList.add(bean);
        bean =new LearnResouce("CSDN——小当博客专栏","Sping Boot学习","http://blog.csdn.net/column/det
        learnList.add(bean);
        bean =new LearnResouce("梁桂钊的博客","Spring Boot 揭秘与实战","http://blog.csdn.net/column/de
        learnList.add(bean);
        bean =new LearnResouce("林祥纤博客系列","从零开始学Spring Boot ","http://412887952-qq-com.iteye
        learnList.add(bean);
        ModelAndView modelAndView = new ModelAndView("/index");
        modelAndView.addObject("learnList", learnList);
        return modelAndView;
    }
```

—— jsp页面编写 ——

```
<body style="background-image: none;">
<div class="body_wrap">
    <div class="container">
        <div class="alert alert-success text-center" role="alert">Sptring Boot学习资源大奉送，爱我就关
        <table class="table table-striped table-bordered">
            <tr>
                <td>作者</td>
                <td>教程名称</td>
                <td>地址</td>
            </tr>
            <c:forEach var="learn" items="${learnList}">
                <tr class="text-info">
                    <td th:text="${learn.author}">嘟嘟MD</td>
                    <td th:text="${learn.title}">SPringBoot干货系列</td>
                    <td><a href="#" th:href="${learn.url}" class="btn btn-search btn-green" target=
                    </td>
                </tr>
            </c:forEach>
        </table>
    </div>
</div>
</body>
```

—— 启动类 ——

启动类不变还是最简单的

```
@SpringBootApplication
public class Application  {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

—— 内嵌Tomcat容器运行项目 ——

基本配置好了就可以启动项目，通过http://localhost:8080/learn 访问，我使用的SpringBoot是1.5.2版本，jdk1.8,以前介绍过，运行项目有三种方式，这里我都做过了一次测试，发现在maven中jasper依赖有加provided和注释掉该依赖范围运行的效果不大一样，具体对比如下：

有添加provided的情况：

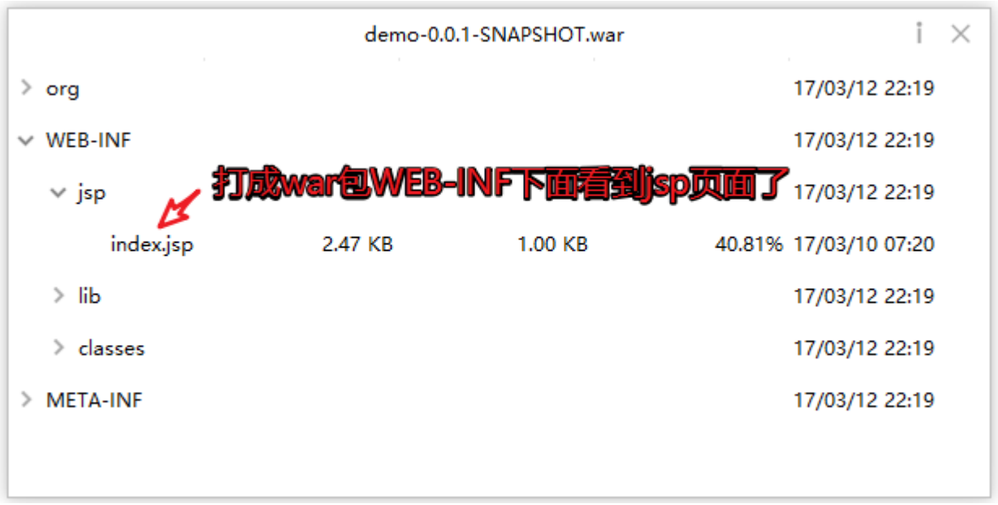
- 右键运行启动类，访问页面报404错误
- 使用spring-boot:run运行正常
- 打包成jar，通过 java -jar demo-0.0.1-SNAPSHOT.jar 运行报错
- 打包成war，通过 java -jar demo-0.0.1-SNAPSHOT.war 运行正常

把provided 注释掉的情况

- 右键运行启动类，访问页面正常
- spring-boot:run运行 访问页面正常
- 打包成jar，通过 java -jar demo-0.0.1-SNAPSHOT.jar 运行报错
- 打包成war，通过 java -jar demo-0.0.1-SNAPSHOT.war 运行正常

我测试了好几次都是这样，就是有加provided的时候，右键运行启动类访问页面的时候，提示404错误。其他3种情况都一样， jar运行也报404，spring-boot:run以及war运行都可以。

为什么jar包运行不行呢，我们打开打包的jar和war分别看看区别，如下2图所示：



从上面可以看出来，jar包运行的时候会404错误，因为默认jsp不会被拷贝进来，而war包里面有包含了jsp，所以没问题。

内嵌Tomcat属性配置

关于Tomcat的偶有属性都在org.springframework.boot.autoconfigure.web.ServerProperties配置类中做了定义，我们只需在application.properties配置属性做配置即可。通用的Servlet容器配置都已”server”左右前缀，而Tomcat特有配置都以”server.tomcat”作为前缀。下面举一些常用的例子。

配置Servlet容器：

```
#配置程序端口，默认为8080
server.port= 8080
#用户绘画session过期时间，以秒为单位
server.session.timeout=
# 配置默认访问路径，默认为/
server.context-path=
```

配置Tomcat：

```
# 配置Tomcat编码,默认为UTF-8
server.tomcat.uri-encoding=UTF-8
# 配置最大线程数
server.tomcat.max-threads=1000
```

更为详细的Servlet容器配置以及Tomcat配置，可以前往博主之前文章查看：Spring Boot干货系列：常用属性汇总

—— 外部的Tomcat服务器部署war包 ——

Spring Boot项目需要部署在外部容器中的时候，Spring Boot导出的war包如果直接在Tomcat的部署会报错，不信你可以试试看。
需要做到下面两点修改才可以：

- 继承SpringBootServletInitializer

外部容器部署的话，就不能依赖于Application的main函数了，而是要以类似于web.xml文件配置的方式来启动Spring应用上下文，此时我们需要在启动类中继承SpringBootServletInitializer并实现configure方法：

```
public class Application extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(Application.class);
    }
}
```

这个类的作用与在web.xml中配置负责初始化Spring应用上下文的监听器作用类似，只不过在这里不需要编写额外的XML文件了。

- pom.xml修改tomcat相关的配置

如果要将最终的打包形式改为war的话，还需要对pom.xml文件进行修改，因为spring-boot-starter-web中包含内嵌的tomcat容器，所以直接部署在外部容器会冲突报错。这里有两种方法可以解决，如下
方法一：

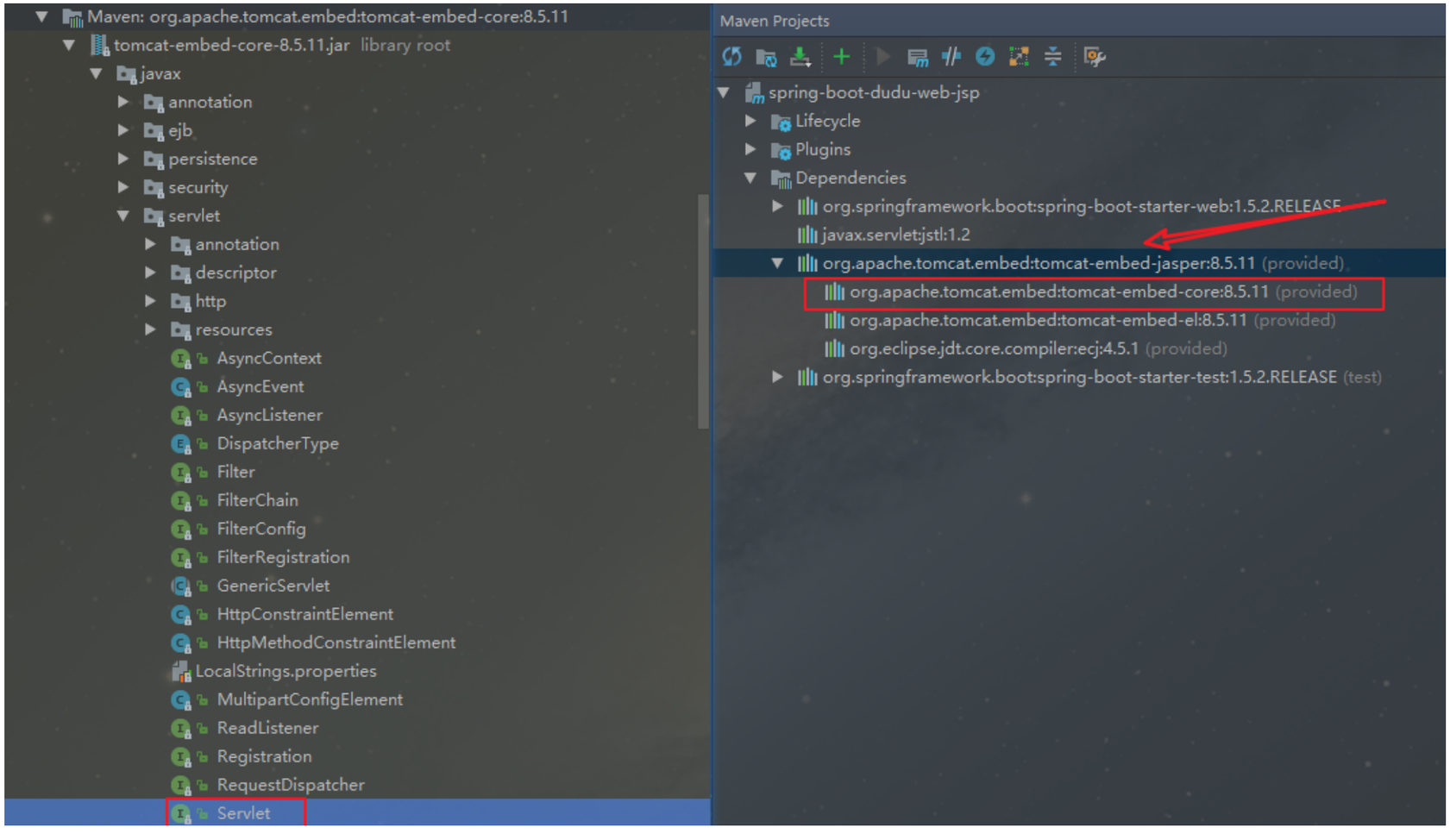
```
<dependency>
```

```
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

在这里需要移除对嵌入式Tomcat的依赖，这样打出的war包中，在lib目录下才不会包含Tomcat相关的jar包，否则将会出现启动错误。
还有一个很关键的关键点，就是tomcat-embed-jasper中scope必须是provided。

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```

因为SpringBootServletInitializer需要依赖 javax.servlet，而tomcat-embed-jasper下面的tomcat-embed-core中就有这个javax.servlet，如果没用provided，最终打好的war里面会有servlet-api这个jar，这样就会跟tomcat本身的冲突了。这个关键点同样适应于下面说的第二种方法。



方法二：

直接添加如下配置即可：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
```

provided的作用上面已经介绍的很透彻了，这里就不啰嗦了，这种方式的好处是，打包的war包同时适合java -jar命令启动以及部署到外部容器中。

如果你不喜欢默认的打包名称，你可以通过节点里添加内容。

```
<build>
  <finalName>springBootJsp</finalName>
</bulid>
```

最后启动tomcat输入http://localhost:8080/springBootJsp/learn 查看效果，还是美美哒

Sptring Boot学习资源大奉送，爱我就关注嘟嘟公众号：嘟爷java超神学堂

作者	教程名称	地址
官方参考文档	Spring Boot Reference Guide	点我
官方SpriongBoot例子	官方SpriongBoot例子	点我
龙国学院	Spring Boot 教程系列学习	点我
嘟嘟MD独立博客	Spring Boot干货系列	点我
后端编程嘟	Spring Boot教程和视频	点我
程序猿DD	Spring Boot系列	点我
纯洁的微笑	Sping Boot系列文章	点我
CSDN——小当博客专栏	Sping Boot学习	点我
梁桂钊的博客	Spring Boot 揭秘与实战	点我
林祥纤博客系列	从零开始学Spring Boot	点我

—— 关于使用jar部署 ——

上面已经测试过了，正常情况下包含jsp的页面是无法用jar的运行的，因为jsp默认是在webapp目录下，可是打包成jar是没有webapp这个目录结构的。

虽然网上有介绍说通过pom.xml配置，把WEB-INF目录复制到META-INF/resources下面。但是博主试了一整天还是访问不了，最后放弃了。各位如何有兴趣可以继续尝试，毕竟war也可以通过java -jar命令来启动的不是么。

总结

我相信全网都找不到一篇有我这篇这么详细的介绍Spring Boot使用jsp的文章。有很多人问我，为什么我的很多文章这么简单易懂，我每次都是哭